Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L1 and (load$ with (application adj1 server$) with programs)

Display: 10 **Documents in Display Format:** KWIC **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Monday, February 09, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L6	L1 and (load$ with (application adj1 server$) with programs)	16	L6
L5	L3	3	L5
L4	L3	3	L4
L3	L1 and (remote$ with load$ with (application adj1 server$))	3	L3
L2	L1 and (load$ with (application adj1 server$))	103	L2
L1	709/$.ccls.	14342	L1

END OF SEARCH HISTORY

L4 / 1, 2, 4

6,687,745 6,453,348

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 3 of 3 returned.

☐ 1. Document ID: US 6687745 B1

L4: Entry 1 of 3 File: USPT Feb 3, 2004

DOCUMENT-IDENTIFIER: US 6687745 B1
TITLE: System and method for delivering a graphical user interface of remote applications over a thin bandwidth connection

Detailed Description Text (113):
Preferably, droplet-enabled applications are deployed as dynamically loadable libraries that, once invoked locally by a client computer, are loaded and run under the control of application drivers on a remote server such as, for example, the application server 40. The droplet-enabled applications and the application drivers are preferably written in the Java, C++, Visual Basic, or other equivalent programming languages that support COM or CORBA interfaces. Also, the droplet-enabled application is implemented as an object that is instantiated at startup, holds all program information while active, and causes the application to terminate when the object is destroyed. By instantiating many such objects within a single application server, the server can serve any number of users running independent instances of the same application from the same server.

Current US Original Classification (1):
709/219

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 2. Document ID: US 6453348 B1

L4: Entry 2 of 3 File: USPT Sep 17, 2002

DOCUMENT-IDENTIFIER: US 6453348 B1
TITLE: Extranet architecture

Brief Summary Text (12):
According to a fourth aspect of the invention there is provided an extranet including a network coupling a plurality of non-related participants wherein each participant is located remotely from the other participants, and a server coupled to the network, the server storing a plurality of applications including workgroup applications, transaction applications, security applications and transport applications wherein the server is programmed to load particular ones of the plurality of applications onto the network for use by the plurality of participants

h e b b g e e e f e ef b e

in response to a request by one of the plurality of participants.

Current US Original Classification (1):
709/225

Current US Cross Reference Classification (1):
709/200

Current US Cross Reference Classification (2):
709/201

Current US Cross Reference Classification (3):
709/202

Current US Cross Reference Classification (4):
709/203

Current US Cross Reference Classification (5):
709/217

Current US Cross Reference Classification (6):
709/218

Current US Cross Reference Classification (7):
709/219

Current US Cross Reference Classification (8):
709/226

Current US Cross Reference Classification (9):
709/227

Current US Cross Reference Classification (10):
709/228

Current US Cross Reference Classification (11):
709/229

Current US Cross Reference Classification (12):
709/230

Current US Cross Reference Classification (13):
709/245

☐ 3. Document ID: US 6038664 A

L4: Entry 3 of 3 File: USPT Mar 14, 2000

DOCUMENT-IDENTIFIER: US 6038664 A
TITLE: Method for selecting communication access method for local area networks

Brief Summary Text (39):
The CP executes a subroutine to determine if the user has access to the requested

h e b b g e e e f e ef b e

program. If the user has access, the program will evaluate the program's preset configuration. The preset configuration may direct the program to run via a particular access method, prompt the user to choose an access method, or evaluate the rules database. If the rules evaluation is directed by the preset configuration, the CP collects information about the requested program and transmits this information along with a request for rule evaluation to the server. The rule evaluation software evaluates the rules and determines which method should be used. If the Remote Control method is selected, a load balancing algorithm may be used to select an application server to host the remote control session.

Current US Cross Reference Classification (1):
709/217

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Term	Documents
3.USPT.	3
(L3).USPT.	3

Display Format: KWIC Change Format

Previous Page Next Page Go to Doc#

h e b b g e e e f e ef b e

Refine Search

Search Results -

Term	Documents
APPLICATION	1839426
APPLICATIONS	927788
PROGRAMS	124225
PROGRAMME	5515
PROGRAMMES	1323
PROGRAM	296405
LOAD$	0
LOAD	542917
LOADA	28
LOADABILITIES	11
LOADABILITY	670
(L1 AND (LOAD$ WITH (APPLICATION ADJ1 SERVER$) WITH PROGRAMS)).USPT.	16

There are more results than shown above. Click here to view the entire set.

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L6

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Monday, February 09, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		

h e b b cg b e e ch

Refine Search

Search Results -

Term	Documents
DESCRIPTOR$	0
DESCRIPTOR	7999
DESCRIPTORADDRESS	1
DESCRIPTORBASE	1
DESCRIPTORCOUNT	1
DESCRIPTORDESCRIPTOR	1
DESCRIPTORE	1
DESCRIPTORED	1
DESCRIPTORENTRY	2
DESCRIPTORFAULT	1
DESCRIPTORIBUFFER	1
(L1 AND (DESCRIPTOR$ WITH ADAPTER$)).USPT.	42

There are more results than shown above. Click here to view the entire set.

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L10

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, February 09, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L10	L1 and (descriptor$ with adapter$)	42	L10

up to 26

h e b b cg b e e ch

L9	L8	1	L9
L8	L1 and (load$ with descriptor$ with adapter$)	1	L8
L7	L2 and (load$ with descriptor$ with adapter$)	0	L7
L6	L4 and (load$ with descriptor$)	0	L6
L5	L4 and (load$ with descriptor$)	0	L5
L4	L3 and L2	2	L4
L3	L1 and (resource adj1 adapter$)	14	L3
L2	L1 and (application adj1 server$)	991	L2
L1	709/$.ccls.	14342	L1

END OF SEARCH HISTORY

First Hit Fwd Refs

Generate Collection

L10: Entry 1 of 42 File: USPT Nov 11, 2003

DOCUMENT-IDENTIFIER: US 6647423 B2
TITLE: Direct message transfer between distributed processes

Detailed Description Text (3):
Referring to FIG. 2, the VI architecture comprises four basic components: virtual interface (VI) 200; VI consumer 202; VI provider 204; and completion queue (CQ) 206. A VI is the mechanism that allows VI consumer 202 to directly access VI provider 204. Each VI represents a communication endpoint, and endpoint pairs may be logically connected to support bi-directional, point-to-point data transfer. Referring now to FIG. 3, VI 200 consists of a pair of work queues; send queue 300 and receive queue 302. VI consumer 202 posts requests, in the form of descriptors, onto either the send queue 300 or the receive queue 302, and removes completed descriptors from either the send 300 or receive queues 302. A descriptor is a memory structure that contains the information VI provider 204 needs to process a request from VI consumer 202, such as pointers to data buffers (see discussion below). VI provider 204 asynchronously processes posted descriptors and marks them with a status value when completed. Each work queue has an associated doorbell that is used to notify the VI network adapter (an element of VI provider 204) that a new descriptor has been posted to a work queue. Doorbells are directly implemented by the adapter and require no operating system (OS) intervention to operate.

Current US Original Classification (1):
709/229

First Hit Fwd Refs

☐ Generate Collection

L10: Entry 4 of 42 File: USPT Aug 20, 2002

DOCUMENT-IDENTIFIER: US 6438613 B1
TITLE: Method and apparatus for allowing packet data to be separated over multiple bus targets

Abstract Text (2):
A method for transferring packet data between a first bus target, a second bus target, and the network adapter comprising the steps of creating a set of descriptor entries in the network adapter, wherein one descriptor entry is generated for each portion of each packet to be transmitted between either the first bus target or the second bus target, and the network adapter; and, transferring a portion of packet data from either the first bus target or the second bus target to the network adapter, wherein the bus target on which the portion of packet data is contained is described by one of the set of descriptor entries.

Brief Summary Text (16):
In operation, the host processor initializes a set of I/O configuration registers inside the network adapter at system initialization to contain the I/O transfer characteristics of each device connected to an I/O bus. During normal transmit operations where the data is being sent to the computer network, host processor would create one descriptor entry for each portion of each packet to be transmitted from the computer. The host processor would then pass the descriptor entries to the network adapter and the network adapter would be responsible for reading the data from the bus target device described by the descriptor entry using the I/O bus interface and burst dispatcher. The data that is read by the I/O bus interface is then passed to the synchronization and buffering logic, which sends the data to the network media interface.

Detailed Description Text (14):
In summary, a packet is described by a set of TX data descriptor entries, where each TX data descriptor entry in the set points to a storage buffer which stores a portion of the data of the packet. As the data comprising each packet can be contained in one or more storage buffers, host processor 3 has to create one or more TX data descriptor entries for each packet that host processor 3 wants network adapter 15 to transmit.

Detailed Description Text (42):
Referring to block 419, if all the descriptor entries have been processed, then network adapter 15 would wait for host processor 3 to place the address of the last descriptor entry in a new set of descriptor entries to storage location 305. Otherwise, if there are more descriptor entries to be processed, then blocks 409 through 419 would be repeated for each descriptor entry and the associated storage buffer to be processed and transmitted, respectively. Thus, if host processor 3 had created 3 TX descriptor entries in block 401, then blocks 409 through 419 would be repeated 3 times--once for each TX descriptor entry to be processed--resulting in the transmission of 3 storage buffers.

Current US Original Classification (1):
709/250

h e b b g e e e f e e g

Current US Cross Reference Classification (1):
709/230

Current US Cross Reference Classification (2):
709/236

Current US Cross Reference Classification (3):
709/246

CLAIMS:

1. A method for transferring packet data between a first bus target and a network adapter in a computer system having an I/O bus, wherein said first bus target is coupled to said I/O bus, a second bus target is coupled to said I/O bus, and said network adapter is coupled to said I/O bus and a network media, comprising the steps of: creating a set of descriptor entries, wherein one descriptor entry is generated for each portion of each packet to be transmitted between one of said first bus target and said second bus target, and said network adapter; and transferring a portion of packet data from one of said first bus target and second bus target to said network adapter wherein the bus target on which said portion of packet data is contained is described by one of said set of descriptor entries.

5. A method for transferring packet data between a bus target and a network adapter in a computer system having an I/O bus, wherein the bus target is coupled to the I/O bus, the network adapter is coupled to the I/O bus, and a packet comprises one or more portions, comprising: creating a set of descriptor entries, wherein one descriptor entry is generated for each portion of the packet, and each portion is to be transmitted between the first bus target and the network adapter; and transferring each portion of the packet from said bus target to the network adapter wherein a location on said bus target on which the portion of packet data is contained is described by one of the set of descriptor entries.

First Hit Fwd Refs

Generate Collection

L10: Entry 9 of 42 File: USPT Apr 3, 2001

DOCUMENT-IDENTIFIER: US 6212567 B1
TITLE: Method and apparatus for performing raw cell status report frequency mitigation on receive in a network node

Detailed Description Text (482):
In reference to the above description, it can be understood that the network node operates in a manner that obviates the need for control reads on the system bus for the normal flow of data in both the transmit and receive directions. The host system writes control information in the form of tx and rx slot descriptors to the adapter. The adapter writes control information in the form of status reports to the host. Therefore, by reducing the system bus bandwidth ordinarily consumed by control reads, overall throughput is greatly improved. In the event that an exception condition occurs, the host system may need to read the CSRs. Because interrupt rates can be mitigated, however, this type of control read should be fairly infrequent.

Current US Original Classification (1):
709/231

First Hit Fwd Refs

Generate Collection

L10: Entry 13 of 42 File: USPT May 23, 2000

DOCUMENT-IDENTIFIER: US 6067563 A
TITLE: Method and apparatus for avoiding control reads in a network node

Brief Summary Text (13):
In one embodiment of the method of the invention, the method performs the step of deciding whether or not to report consumption of a tx slot corresponding to a raw cell. If the consumption of a tx slot corresponding to the raw cell is to be reported, then the method performs the step of posting the tx slot by writing a tx slot descriptor associated with the tx slot to the adapter, setting an EOP field in the tx slot descriptor to one. In response to the posting of the tx slot, a data transfer request corresponding to each portion of data to be read from the tx slot in host memory in a separate data transfer operation is created. If the data portion is the last in the tx slot, then the method determines if the EOP field in the tx slot descriptor associated with the posted transmit slot is set. If the EOP field is set to one, then the data transfer request is serviced and a status report indicating slot consumption activity is sent to the host system.

Detailed Description Text (492):
In reference to the above description, it can be understood that the network node operates in a manner that obviates the need for control reads on the system bus for the normal flow of data in both the transmit and receive directions. The host system writes control information in the form of tx and rx slot descriptors to the adapter. The adapter writes control information in the form of status reports to the host. Therefore, by reducing the system bus bandwidth ordinarily consumed by control reads, overall throughput is greatly improved. In the event that an exception condition occurs, the host system may need to read the CSRs. Because interrupt rates can be mitigated, however, this type of control read should be fairly infrequent.

Current US Original Classification (1):
709/212

CLAIMS:

1. A method of reporting information to a host system connected to a data interface by an adapter, the host system having a host memory including a plurality of memory slots to store data to be transmitted to the data interface, wherein the data includes a plurality of data units, the method comprising:

determining whether consumption of a memory slot is to be reported;

posting the memory slot by sending a slot descriptor of the memory slot from the host system to the adapter, the slot descriptor including an indication of whether to report the consumption of the memory slot;

initiating a sequential transmission of data units from the memory slot to the adapter; and

reporting the consumption of the memory slot to the host system when a last data unit from the memory slot is to be transmitted and the slot descriptor indicates

that consumption of the memory slot is to be reported.

First Hit Fwd Refs

Generate Collection

L10: Entry 19 of 42 File: USPT Aug 17, 1999

DOCUMENT-IDENTIFIER: US 5940404 A
TITLE: Method and apparatus for enhanced scatter mode allowing user data to be page aligned

Drawing Description Text (4):
FIG. 2 is a diagram illustrating a logical channel descriptor or connector setup information and chip wide information used by the adapter of the preferred embodiment;

Drawing Description Text (9):
FIG. 5 is a diagram illustrating a DMA descriptor address queue for DMA descriptors DMAed to a PCI system buffer by the adapter of the preferred embodiment;

Detailed Description Text (11):
FIG. 5 illustrates a DMA descriptor block or chain 502 pointed to by a DMA descriptor address queue 504. Each DMA descriptor 502 includes flags, length (LEN), a source address (SRC) and a destination address (DEST). As shown in FIG. 5, the DMA descriptor flags, length (LEN), source address (SRC) and destination address (DEST) for the first DMA descriptor are built by the adapter 100 to point to the packet header 322 of the current receive buffer 320. The length (LEN) includes the packet header 322, DMA list 324, and the protocol header bytes (NUM.sub.-- HEAD.sub.-- BYTES) 406. The packet header 322, DMA descriptor block 502 and the protocol header bytes (NUM.sub.-- HEAD.sub.-- BYTES) 406 are DMAed to a separate system buffer 510 in the PCI memory address space 114.

Current US Cross Reference Classification (1):
709/212

CLAIMS:

10. An adapter for providing enhanced scatter mode allowing user data to be page aligned in a memory as recited in claim 9 wherein said means for performing said single page mode transfer of said identified small packet to the memory includes means for building a DMA descriptor for said identified small packet and means for enqueuing said DMA descriptor to a DMA engine to execute said DMA descriptor.

11. An adapter for providing enhanced scatter mode allowing user data to be page aligned in a memory as recited in claim 8 wherein said means for transferring said first scatter page containing said packet header, said DMA list and said protocol header bytes includes means for building a DMA descriptor for said first scatter page and means for enqueuing said DMA descriptor to a DMA engine to execute said DMA descriptor.

h e b b g e e e f e e g

First Hit Fwd Refs

Generate Collection

L10: Entry 20 of 42 File: USPT Aug 3, 1999

DOCUMENT-IDENTIFIER: US 5931915 A
TITLE: Method for processing early arrival messages within a multinode asynchronous data communications system

Detailed Description Text (6):
With reference to FIG. 2, the receiving node, under control of an application program, allocates a section of system memory for a particular message to be received from another computing node in the system. This receive buffer is "posted" when a command is given to the communications adapter to tell the adapter where to store the message once received. A buffer poster 42 within communications adapter 20 stores the descriptor information about the allocated receive buffer into a control store memory 46 via a memory controller 44. This descriptor information is stored in memory 46 until the corresponding message arrives. These descriptors in essence comprise pointers to the real receive buffers which have been posted in main system memory 26 (FIG. 1).

Current US Original Classification (1):
709/232

Current US Cross Reference Classification (1):
709/237

Current US Cross Reference Classification (2):
709/250

First Hit Fwd Refs

Generate Collection

L10: Entry 23 of 42 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5872956 A
TITLE: Design methodology for device drivers supporting various operating systems network protocols and adapter hardware

Abstract Text (1):
An alterable device driver development system for supporting various operating systems, network protocols, and adapter hardware interfaces. Each device driver comprises a System, Network and Adapter software component, each component providing services to the other through its associated programming interfaces and being alterable or replaceable according to the requirements of an associated adapter hardware product. The System component supports a set of services defined by the device driver and which can be used by the Network and Adapter components. The Network component manages all interactions of the Adapter component with a network operating system or protocol stack and ensures its applicability and correctness in the context of the device driver operation. The Adapter component provides the functions for operating and managing all interactions with the adapter hardware. A Transformation Path in the Adapter component improves device driver performance by mapping network packet descriptors against adapter packet descriptors in data transferring processes.

Brief Summary Text (21):
Another object is a multiple component device driver model architecture including a performance enhancement path which allows direct mapping between a network protocol stack of packet descriptors and hardware packet descriptors for critical receive and transmit operations within an adapter component.

Brief Summary Text (23):
These and other objects, features and advantages are achieved in a data processing system including LANs and/or WANs in which a device driver design methodology uses at least three main components to enable a developer to select and combine key components to support a wide variety of operating systems and platforms, network protocols and adapter hardware. Each device driver is comprised of a System, Network and Adapter software component. Each component provides services to the other through an associated programming interface and is alterable or replaceable according to the requirements of the operating system or platform, network protocol or adapter hardware. The System component supports the services defined by the device driver and operating system which can be used by the Network and Adapter component. The Network component manages all interactions of the Adapter component with network operating systems and insures its applicability and correctness in the context of the device driver operation. The Adapter component provides the function for operating and managing all interactions with the adapter hardware. A Transformation Path in the Adapter component improves the translation of network protocol stack descriptors against adapter packet descriptors in data transferring processes.

Detailed Description Text (31):
The Transformation Path 118 added to the present design has a feature to address performance costs. In its architectural sense, the Transformation Path is a specialized part of the Adapter software component that has detailed knowledge of a Network component's transmit and receive packet descriptor format as specified by

its network protocol stack. Each network interface defines a data structure to be used for building a packet to be transmitted onto the LAN/WAN and another data structure for describing a packet received from the LAN/WAN. These packet data structures, commonly referred to in the industry as descriptors or fragment lists, are formatted differently among the different network interfaces. Likewise, each type of adapter hardware defines what it needs to know about packets in order to transmit and receive them on the attached physical network. Thus, the various Network component packet descriptor formats need to be mapped to a particular hardware packet descriptor format and vice versa. The role of the Transformation Path, therefore, is to perform this descriptor mapping within the Adapter component in a Network component-specific fashion. Doing so saves the costly overhead of using an intermediate generic descriptor format to convert between the different packet descriptor types for transmission and reception operations.

Detailed Description Text (37):
It is important to understand that the Transformation Path is defined by the device driver model architecture as a specialized part of the Adapter component's transmit and receive path whose purpose is to supply the mapping between the different Network component protocol packet descriptors and the packet descriptor format required by the adapter hardware.

Current US Original Classification (1):
709/224

CLAIMS:

1. A design methodology for a device driver supporting a wide variety of system platforms and operating system; network protocols and adapter hardware, comprising the steps of:

forming a System, Network and Adapter component in the device driver;

defining a set of services for each component whereby such services can be used by the other components;

defining a program interface between each one of the components;

defining the public data supported by each component;

forming a Transformation Path in the adapter component for mapping network and adapter hardware descriptors in data processing operations; and

creating or updating component source code to form new device driver executable code to support a new operating system or network protocol or adapter hardware.

3. The data processing system of claim 2 further including a Transformation Path in the Adapter component for mapping network packet descriptors against adapter packet descriptors in data transferring processes directed by the device driver.

9. The method of claim 8 further comprising the step of:

installing a Transformation Path in the Adapter component for mapping network packet descriptors against adapter packet descriptors in data transferring processes directed by the device driver.

11. The article of manufacture of claim 10 further including computer readable code means for installing a Transformation Path in the Adapter component for mapping network packet descriptors against adapter packet descriptors in data transferring processes directed by the device driver.

First Hit Fwd Refs

Generate Collection

L10: Entry 26 of 42 File: USPT Aug 11, 1998

DOCUMENT-IDENTIFIER: US 5793953 A
TITLE: Method and apparatus for allowing packet data to be separated over multiple bus targets

Brief Summary Text (17):
In operation, the host processor initializes a set of I/O configuration registers inside the network adapter at system initialization to contain the I/O transfer characteristics of each device connected to an I/O bus. During normal transmit operations where the data is being sent to the computer network, host processor would create one descriptor entry for each portion of each packet to be transmitted from the computer. The host processor would then pass the descriptor entries to the network adapter and the network adapter would be responsible for reading the data from the bus target device described by the descriptor entry using the I/O bus interface and burst dispatcher. The data that is read by the I/O bus interface is then passed to the synchronization and buffering logic, which sends the data to the network media interface.

Detailed Description Text (15):
In summary, a packet is described by a set of TX data descriptor entries, where each TX data descriptor entry in the set points to a storage buffer which stores a portion of the data of the packet. As the data comprising each packet can be contained in one or more storage buffers, host processor 3 has to create one or more TX data descriptor entries for each packet that host processor 3 wants network adapter 15 to transmit.

Detailed Description Text (46):
Referring to block 419, if all the descriptor entries have been processed, then network adapter 15 would wait for host processor 3 to place the address of the last descriptor entry in a new set of descriptor entries to storage location 305. Otherwise, if there are more descriptor entries to be processed, then blocks 409 through 419 would be repeated for each descriptor entry and the associated storage buffer to be processed and transmitted, respectively. Thus, if host processor 3 had created 3 TX descriptor entries in block 401, then blocks 409 through 419 would be repeated 3 times--once for each TX descriptor entry to be processed--resulting in the transmission of 3 storage buffers.

Current US Original Classification (1):
709/250

Current US Cross Reference Classification (2):
709/230

Current US Cross Reference Classification (3):
709/246

L6	L1 and (load$ with (application adj1 server$) with programs)	16	L6
L5	L3	3	L5
L4	L3	3	L4
L3	L1 and (remote$ with load$ with (application adj1 server$))	3	L3
L2	L1 and (load$ with (application adj1 server$))	103	L2
L1	709/$.ccls.	14342	L1

END OF SEARCH HISTORY

L6/8, 9, 1

6,301,601 6,292,803

h e b b cg b e e ch

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 10 of 16 returned.

☐ 1. Document ID: US 6643690 B2

L6: Entry 1 of 16 File: USPT Nov 4, 2003

DOCUMENT-IDENTIFIER: US 6643690 B2
TITLE: Apparatus and method for determining a program neighborhood for a client node in a client-server network

Detailed Description Text (59):
The user of the client node 20 can select and launch one of the application programs displayed in the Program Neighborhood window. When launching an application, the Program Neighborhood application can execute the application on the same server 30', where applicable, taking into account load balancing requirements among servers and the availability of the application on that server 30'. The PNAPI 52' can include a launch mechanism for launching a remote application locally on the server 30' when the server 30' is nominated to launch the application. When a different server is needed to run the application, the Program Neighborhood application can launch the application via the server 30' (i.e., server-based client) using the windows to present the application on the desktop of the client node 20 as described above in FIG. 3B.

Current US Original Classification (1):
709/217

Current US Cross Reference Classification (1):
709/203

Current US Cross Reference Classification (2):
709/219

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw De

☐ 2. Document ID: US 6591300 B1

L6: Entry 2 of 16 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591300 B1
TITLE: Integrated management application

Detailed Description Text (5):
The user entry/access system 102 may be, for example, a windows-based PC connected

h e b b g e e e f e ef b e

to a Local Area Network (LAN) and having web-browser software, such as an Java-enabled web browser. The application servers 112 store the IMA program and a plurality of sub-applications described in more detail below. In one exemplary embodiment, the application servers 112 store: an event planning sub-application; an action item sub-application; a contact sub-application; a shared document sub-application; a memorandum sub-application; a resource management sub-application; and a services sub-applications. The IMA and individual sub-applications may be embodied as program code which is loaded on the application servers 112 from a tangible media, such as a diskette or CD-ROM, or via transmission over a network connection.

Current US Original Classification (1):
709/226

Current US Cross Reference Classification (3):
709/202

Current US Cross Reference Classification (4):
709/220

Current US Cross Reference Classification (5):
709/223

Current US Cross Reference Classification (6):
709/224

☐ 3. Document ID: US 6591295 B1

L6: Entry 3 of 16 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591295 B1
TITLE: Methods and apparatus for using multimedia data stored in a relational database in web applications

Brief Summary Text (9):
In a principal aspect, the present invention takes the form of novel interface between Web based programs and a relational database. In accordance with the invention, multimedia objects stored in a relational database are identified by a Universal Resource Locator (URL) taking the form of a character string consisting of (a) an identification of the domain name and port of a host computer connected to the Internet, (b) the identification of a Web agent program which may be dynamically loaded or statically linked into the web server's or application server's environment and which operates as an interface to a relational database, (c) the specification of a specific SQL procedure for storing, retrieving or updating a particular data object, and (d) additional parameter information for specifying the row and column location of the object in a database table and any additional information needed by the SQL procedure for manipulating the object.

Current US Original Classification (1):
709/217

Current US Cross Reference Classification (4):
709/203

h e b b g e e e f e ef b e

Current US Cross Reference Classification (5):
709/219

Current US Cross Reference Classification (6):
709/226

☐ 4. Document ID: US 6539445 B1

L6: Entry 4 of 16 File: USPT Mar 25, 2003

DOCUMENT-IDENTIFIER: US 6539445 B1
**** See image for Certificate of Correction ****
TITLE: Method for load balancing in an application server system

Detailed Description Text (7):
The application server system determines the time it would take to run a job of a slave computer ("execution time") based on review of actual statistics from other jobs for the same application program. In one embodiment, the application server system tracks for each application program the actual execution time of its jobs, the slave computer on which the job ran, and the computing resource load of the slave computer while the job ran. The application server system assumes that similar jobs, that is jobs for the same application program with the same size, will have the same execution time if the job is run in a similar slave environment, that is on the same slave computer with the same computing resource load. Thus, if the actual execution time in a similar slave environment is available for such a similar job, then the application server system uses the actual execution time as an estimate. When such statistics are not available for a similar job that ran in a similar slave environment, the application server system estimates the execution time based on the actual execution time of similar jobs, but in different slave environments. For example, if the actual execution time for a job was 10 seconds with a computing resource load of 50 percent, then the application server system may estimate the execution time of a similar job to be 8 seconds when the computing resource load is 25 percent.

Detailed Description Text (8):
The application server system determines the computing resource load of a slave computer based on the application program loads of the application programs that will be executing at the time. In one embodiment, the application server system identifies which application programs will be executing at the time and then totals the application program loads of the application programs. For example, if 3 jobs will be running at the time and 2 jobs are for an application program with an application program load of 10 and the other job is for an application program with an application program load of 5, then the computing resource load at that time will be 25.

Current US Cross Reference Classification (1):
709/208

Current US Cross Reference Classification (2):
709/209

Current US Cross Reference Classification (3):

h e b b g e e e f e ef b e

709/210

Current US Cross Reference Classification (4):
709/211

☐ 5. Document ID: US 6502148 B1

L6: Entry 5 of 16 File: USPT Dec 31, 2002

DOCUMENT-IDENTIFIER: US 6502148 B1
**** See image for Certificate of Correction ****
TITLE: System for scaling an application server system

Detailed Description Text (6):
The application server system determines the time it would take to run a job of a slave computer ("execution time") based on review of actual statistics from other jobs for the same application program. In one embodiment, the application server system tracks for each application program the actual execution time of its jobs, the slave computer on which the job ran, and the computing resource load of the slave computer while the job ran. The application server system assumes that similar jobs, that is jobs for the same application program with the same size, will have the same execution time if the job is run in a similar slave environment, that is on the same slave computer with the same computing resource load. Thus, if the actual execution time in a similar slave environment is available for such a similar job, then the application server system uses the actual execution time as an estimate. When such statistics are not available for a similar job that ran in a similar slave environment, the application server system estimates the execution time based on the actual execution time of similar jobs, but in different slave environments. For example, if the actual execution time for a job was 10 seconds with a computing resource load of 50 percent, then the application server system may estimate the execution time of a similar job to be 8 seconds when the computing resource load is 25 percent.

Detailed Description Text (7):
The application server system determines the computing resource load of a slave computer based on the application program loads of the application programs that will be executing at the time. In one embodiment, the application server system identifies which application programs will be executing at the time and then totals the application program loads of the application programs. For example, if 3 jobs will be running at the time and 2 jobs are for an application program with an application program load of 10 and the other job is for an application program with an application program load of 5, then the computing resource load at that time will be 25.

Current US Cross Reference Classification (1):
709/208

Current US Cross Reference Classification (2):
709/209

Current US Cross Reference Classification (3):
709/210

Current US Cross Reference Classification (4):
709/211

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 6. Document ID: US 6457047 B1

L6: Entry 6 of 16 File: USPT Sep 24, 2002

DOCUMENT-IDENTIFIER: US 6457047 B1
**** See image for Certificate of Correction ****
TITLE: Application caching system and method

Detailed Description Text (9):
A large number of applications can benefit from distributed application caching in accordance with the invention. For example, as a rapidly increasing number of handheld devices (or small-form-factor devices such as, PDAs, Palm Pilots, pagers, cellular phones) access the web, document summarization becomes very important because of the small-size screen on the handheld devices and the limited bandwidth between handheld devices and web servers using Wireless Application Protocol (WAP). The distributed application caching in accordance with the invention is well suitable for dealing with this problem. In particular, a document summarization program (such as the one from Verity, Inc.) can be cached on the application cache servers across the network. Thus, when an application server receives a request originated from a handheld device, it uses the cached document summarizer to serve up a small-footprint version of the document in the WML (Wireless Markup Language) format, unbeknown to the content originator. Other examples of applications that benefit from the application caching in accordance with the invention include a CGI program or a servlet for Mortgage calculation that can be cached on the application cache servers across the network to reduce the load on the master application server. Now, a method for handling dynamic requests in accordance with the invention will be described.

Current US Original Classification (1):
709/217

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 7. Document ID: US 6362836 B1

L6: Entry 7 of 16 File: USPT Mar 26, 2002

DOCUMENT-IDENTIFIER: US 6362836 B1
**** See image for Certificate of Correction ****
TITLE: Universal application server for providing applications on a variety of client devices in a client/server network

Current US Cross Reference Classification (1):
709/207

h e b b g e e e f e ef b e

CLAIMS:

1. In a client server network comprised of at least one application server providing at least one application program for selection by a user via a client device, a method for controlling the instantiation of application programs provided on an application server to a requesting client device using an universal application server in communication with the application server and the client device, comprising: at the universal application server: providing a datastore comprising information for each user including application passwords and authorized applications, application server location and connection parameters, application program protocols and connection parameters, a bootstrap applet, a login applet, at least one protocol engine, and at least one display engine; establishing a connection between the universal application server and the client device; on receiving a request from the client device, retrieving the bootstrap applet from the datastore and downloading it to the requesting client device; at the client device: running the bootstrap applet on the client device; obtaining user information and client device type and transmitting the user information and client device type to the universal application server; at the universal application server: on receiving the user information, creating a user webtop containing icons associated with the application programs authorized for the user; and downloading the user webtop to the requesting client device for display; waiting for the user to select at least one icon on the user webtop and request the associated application program, on receiving the application program request: performing load balancing to determine which application server to use for the requested application program; instantiating the appropriate protocol engine for the requested application program type; downloading to the requesting client device the display engine for the client device type for execution on the client device, the session manager retrieving the appropriate protocol engine and display engine from the data store, establishing a connection between the instantiated protocol engine and the application server and instantiating an instance of the requested application program on the application server; the protocol engine converting the application display requests to a form suitable for display on the display engine; at the client device: executing the downloaded display engine; establishing a connection between the display engine and the instantiated protocol engine for receiving converted display requests from the requested application program and providing user interaction and requests to the requested application program via the protocol engine; and, at the universal application server: determining if the requested application program is resumable in the event that the connection to the client device is broken, and, if the application is not resumable, disconnecting from the application server and the instance of application program upon exiting of the user, and, if the application program is resumable, on disconnection of the client device, maintaining the connection to the application server and suspending the instance of the application until the client device and user reconnects.

2. The method of claim 1 wherein the step of performing load balancing further comprises: determining which application servers in the network are capable of providing the requested application program; and selecting from the application server determined in the previous step the one having the fewest number of applications running.

33. In a client server network comprised of at least one application server providing at least one application program for selection by a user via a client device, a method for controlling the instantiation of application programs provided on an application server to a requesting client device using an universal application server in communication with the application server and the client device, comprising: at the universal application server: providing a datastore comprising information for each user including application passwords and authorized applications, application server location and connection parameters, application protocols and connection parameters, a bootstrap applet, a login applet, at least

one protocol engine, and at least one display engine; initializing a data store engine, a session manager engine and a web server; establishing a connection between the web server and the client device; on receiving a request from the client device, the web server retrieving the bootstrap applet from the datastore and downloading it to the requesting client device; at the client device: running the bootstrap applet on the client device; obtaining login information including client device type and user information, and transmitting the login information to the web server for approval processing; at the web server: on receiving the login information passing it to the data store engine for approval processing; on not receiving approval from the data store engine, rejecting the login information; on receiving approval from the data store engine, creating a user webtop containing icons associated with the application programs authorized for the user; and downloading the user webtop to the requesting client device for display; waiting for the user to select at least one icon on the user webtop and request the associated application program, and, when the application program request is received from the client device passing the application program request to the session manger engine for processing; at the session manager: performing load balancing to determine which application server to use for the requested application program; instantiating the appropriate protocol engine for the requested application program type; downloading to the requesting client device the display engine for that client device type for execution on the client device, the session manager retrieving the appropriate protocol engine and display engine from the data store, having the instantiated protocol engine establish a connection to the application server and instantiating an instance of the requested application program on the application server; the protocol engine converting the application display requests to a form suitable for display on the display engine; at the client device: executing the display engine; establishing a connection between the display engine and the instantiated protocol engine for receiving converted display requests from the requested application program and providing user interaction and requests to the requested application program via the protocol engine; and, at the instantiated protocol engine: on receiving user interaction and requests having the protocol engine provide them to the application program for processing; and determining if the requested application program is resumable in the event that the connection to the client device is ended, and, if the application program is not resumable, disconnecting from the application server and the instance of application program upon exiting of the user, and, if the application program is resumable, on disconnection of the client device maintaining the connection to the application server and suspending the instance of the application program until the client device and user reconnects.

34. The method of claim 33 wherein the step of performing load balancing further comprises: determining which application servers in the network are capable of providing the requested application program; and selecting from the application server determined in the previous step the one having the fewest number of applications running.

☐ 8. Document ID: US 6301601 B1

L6: Entry 8 of 16 File: USPT Oct 9, 2001

DOCUMENT-IDENTIFIER: US 6301601 B1
TITLE: Disabling and enabling transaction committal in transactional application

h e b b g e e e f e ef b e

components

Detailed Description Text (15):
The illustrated transaction server executive 80 is implemented as a dynamic link library ("DLL"). (A DLL is a well-known executable file format which allows dynamic or run-time linking of executable code into an application program's process.) The transaction server executive 80 is loaded directly into application server processes (e.g., "ASP" 90) that host server application components, and runs transparently in the background of these processes.

Current US Cross Reference Classification (2):
709/203

Current US Cross Reference Classification (3):
709/223

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 9. Document ID: US 6292800 B1

L6: Entry 9 of 16 File: USPT Sep 18, 2001

DOCUMENT-IDENTIFIER: US 6292800 B1
TITLE: Database system

Detailed Description Text (9):
FIG. 3 shows additional details of ADB switcher 211. ADB switcher 211 includes request queuing and forwarding logic 311. An ADB switcher may also include one or more application server-specific message processing libraries. For example, the ADB switcher 211 is shown having two message processing libraries 331 and 332. Each library 331 and 332 can include dynamically loaded program code that implements functions used to process data requests from particular application servers, interact with database systems 231-234, and generate replies to particular application servers 201 or 202. Functions implemented by libraries 331 and 332 may include queuing request messages from application servers 201-202, generating database system 231-234 queries based on received request messages, and determining particular database systems 231-234 to receive queries. Libraries 331-332 may also receive query responses from the database systems 231-234, format responses to the application servers 201-202 based on the query responses, and send the responses to the application servers 201-202.

Current US Cross Reference Classification (2):
709/201

Current US Cross Reference Classification (3):
709/227

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 10. Document ID: US 6141759 A

h e b b g e e e f e ef b e

L6: Entry 10 of 16 File: USPT Oct 31, 2000

DOCUMENT-IDENTIFIER: US 6141759 A
TITLE: System and architecture for distributing, monitoring, and managing information requests on a computer network

Detailed Description Text (10):
The configuration information is stored on the server computer system's data storage device 34 and loaded into memory, such as the server computer system's RAM 36, when the Request Broker 90 is initialized by the web server program 64, or when the Application Server program 92 is initialized by its application server computer system 72, 74, or web server program 64.

Current US Cross Reference Classification (1):
709/203

Term	Documents
APPLICATION	1839426
APPLICATIONS	927788
PROGRAMS	124225
PROGRAMME	5515
PROGRAMMES	1323
PROGRAM	296405
LOAD$	0
LOAD	542917
LOADA	28
LOADABILITIES	11
LOADABILITY	670
(L1 AND (LOAD$ WITH (APPLICATION ADJ1 SERVER$) WITH PROGRAMS)).USPT.	16

There are more results than shown above. Click here to view the entire set.

Display Format: KWIC Change Format

Previous Page Next Page Go to Doc#

h e b b g e e e f e ef b e

L7	(utility adj1 Java adj1 class$)	0	L7
L6	((utility adj1 Java adj1 class$) or(native adj1 librar$))	38	L6
L5	((utility adj1 Java adj1 class$) and (native adj1 librar$))	0	L5
L4	L1 and ((utility adj1 Java adj1 class$) and (native adj1 librar$))	0	L4
L3	L1 and ((utility adj1 Java adj1 class$) with (native adj1 librar$))	0	L3
L2	L1 and ((load$ or package$) with (utility adj1 Java adj1 class$) with (native adj1 librar$))	0	L2
L1	179/$.ccls.	0	L1

END OF SEARCH HISTORY

h e b b cg b e e ch